Mini-Project
Khasa Gillani

1. Self-Learning :

Resources from where I have been studying about smart contract and solidity language:

- https://cryptozombies.io/
- https://www.tutorialspoint.com/solidity/solidity_constructors.htm
- https://www.trufflesuite.com/docs/truffle/overview

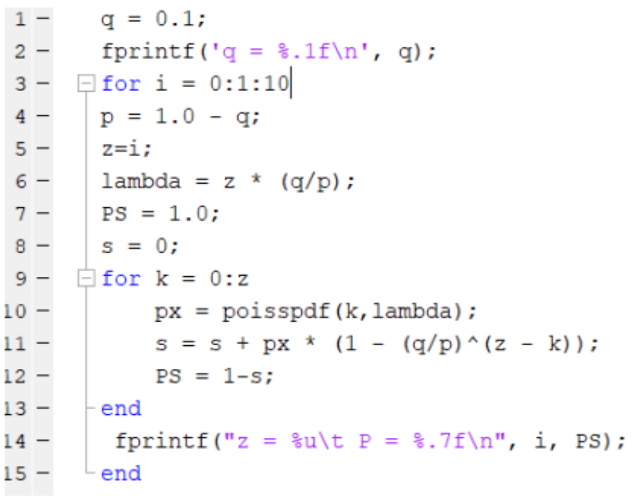
What I have learnt so far:

- The fundamentals of solidity language
- How to develop and deploy smart contracts
- Getting hand on experience by going through zombie factory smart contract and other small unit tests

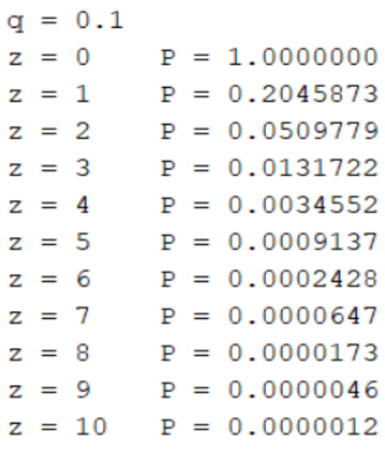
Apart from that I have been doing certification of Blockchain on https://www.coursera.org/ and enrolled in different courses:

- Blockchain Basics
- IBM Blockchain Foundation for Developers

Reason of doing these certifications is to get more expertise and developing understanding of Blockchain and how it works.For getting more insight of foundation of Blockchain I have studied academic literature and implemented the concept of Bitcoin white paper https://bitcoin.org/bitcoin.pdf in MATLAB. Below I have attached the sample code:

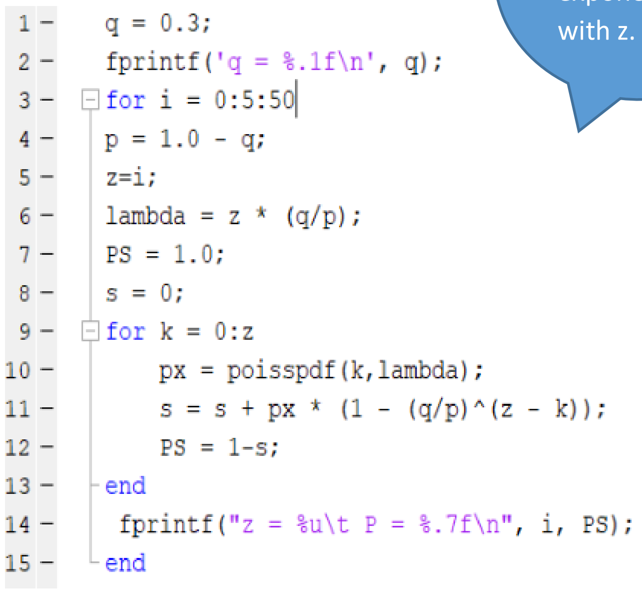


```
q = 0.1;
fprintf('q = %.1f\n', q);
for i = 0:1:10
p = 1.0 - q;
z=i;
lambda = z * (q/p);
PS = 1.0;
s = 0;
for k = 0:z
    px = poisspdf(k,lambda);
    s = s + px * (1 - (q/p)^(z - k));
    PS = 1-s;
end
 fprintf("z = %u\t P = %.7f\n", i, PS);
end
```

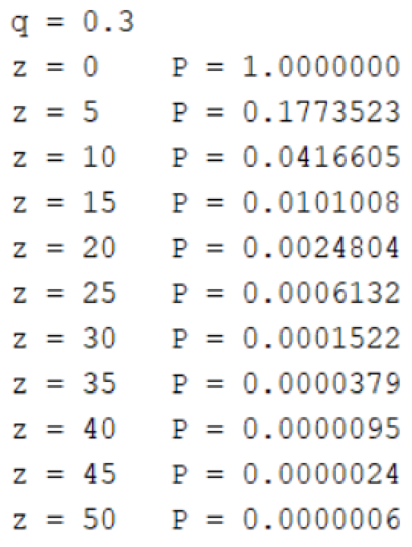


```
q = 0.1
z = 0     P = 1.0000000
z = 1     P = 0.2045873
z = 2     P = 0.0509779
z = 3     P = 0.0131722
z = 4     P = 0.0034552
z = 5     P = 0.0009137
z = 6     P = 0.0002428
z = 7     P = 0.0000647
z = 8     P = 0.0000173
z = 9     P = 0.0000046
z = 10    P = 0.0000012
```

probability drop off exponentially with z.

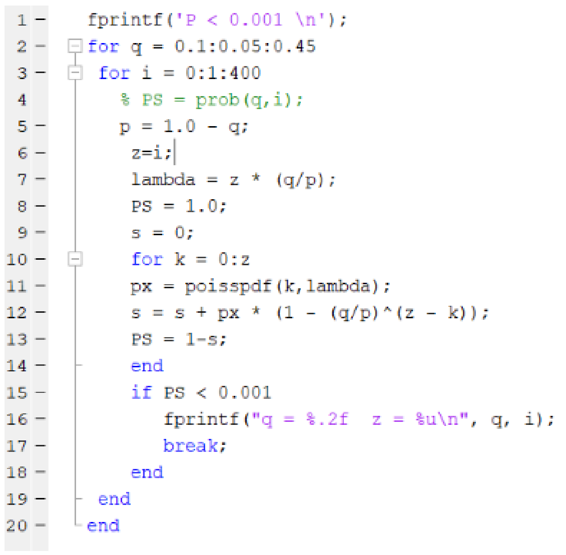


```
q = 0.3;
fprintf('q = %.1f\n', q);
for i = 0:5:50
p = 1.0 - q;
z=i;
lambda = z * (q/p);
PS = 1.0;
s = 0;
for k = 0:z
    px = poisspdf(k,lambda);
    s = s + px * (1 - (q/p)^(z - k));
    PS = 1-s;
end
 fprintf("z = %u\t P = %.7f\n", i, PS);
end
```

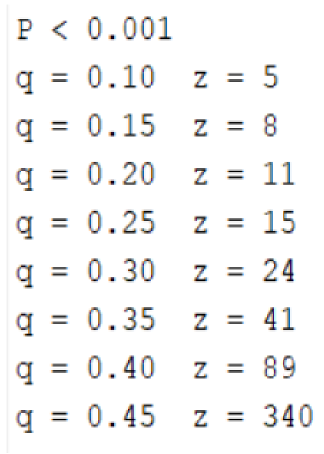


```
q = 0.3
z = 0     P = 1.0000000
z = 5     P = 0.1773523
z = 10    P = 0.0416605
z = 15    P = 0.0101008
z = 20    P = 0.0024804
z = 25    P = 0.0006132
z = 30    P = 0.0001522
z = 35    P = 0.0000379
z = 40    P = 0.0000095
z = 45    P = 0.0000024
z = 50    P = 0.0000006
```

Solving for P < 0.001:



```
fprintf('P < 0.001 \n');
for q = 0.1:0.05:0.45
 for i = 0:1:400
   % PS = prob(q,i);
   p = 1.0 - q;
    z=i;
    lambda = z * (q/p);
    PS = 1.0;
    s = 0;
    for k = 0:z
    px = poisspdf(k,lambda);
    s = s + px * (1 - (q/p)^(z - k));
    PS = 1-s;
    end
    if PS < 0.001
       fprintf("q = %.2f  z = %u\n", q, i);
       break;
    end
 end
end
```



```
P < 0.001
q = 0.10  z = 5
q = 0.15  z = 8
q = 0.20  z = 11
q = 0.25  z = 15
q = 0.30  z = 24
q = 0.35  z = 41
q = 0.40  z = 89
q = 0.45  z = 340
```

After learning through these platforms, I am confident enough to work on blockchain projects because now I can say I have essential knowledge for getting started. However still need long way to get more expertise.

2. Token Exchange Overview

1. ERC20

This smart contract comprises of functions for exchange of ERC tokens between various accounts. It includes the following

- ***Functions***:
    -
    - ***Total Supply***: Return the total number of tokens from associated account. Such as, the subject who has deployed the account "CryptoPayment" is the owner of account.
    - SetToken: Set the number of tokens and owner account address value

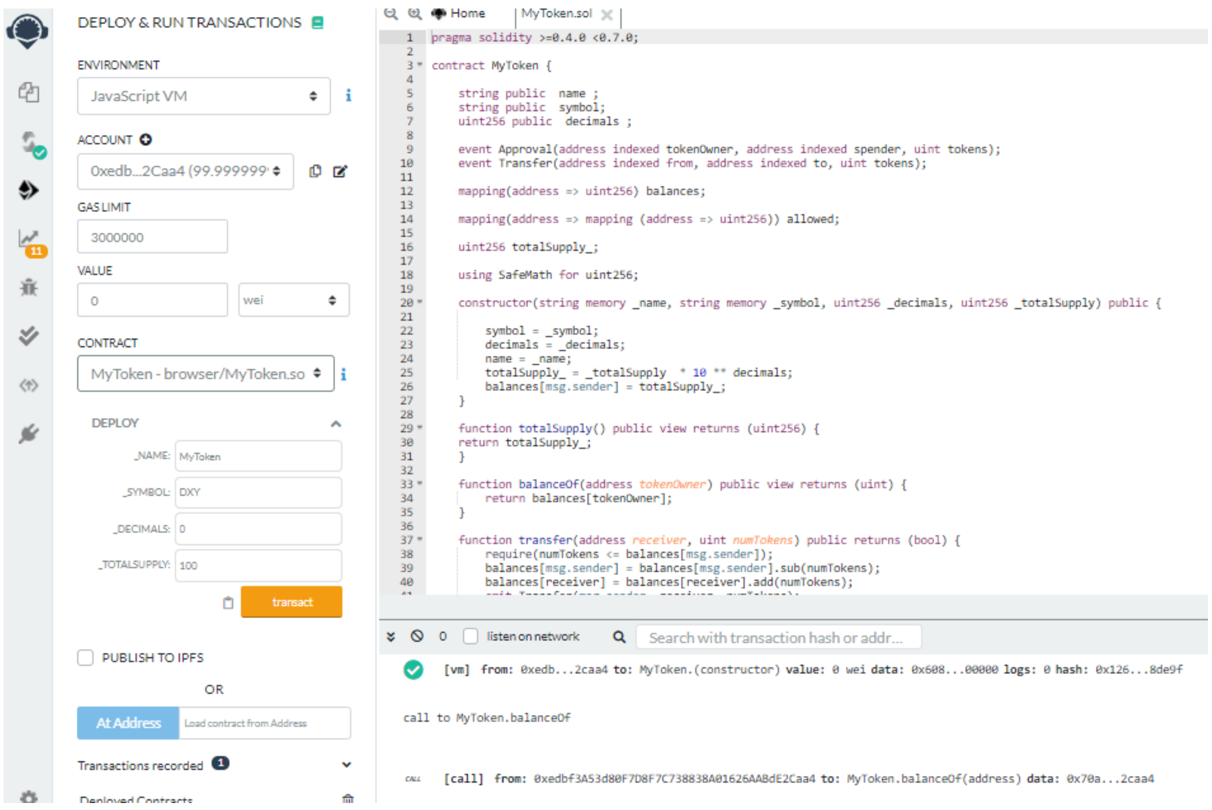
    - ***transfer***: Transfer given token from owner 'account to recipient's account
    - TransferFrom: Transfer given token from mentioned sender account to recipient account
    - Allowance: check transfer permission of tokens from one account to another
    - Approve: set the number of token allowed for transfer from account A to account B
    - balanceOf: Return balance in form of token in given account
- ***Events***:
    - Two events ***Transfer*** and ***Approve*** are declared which get to be triggered upon transfer of token and approval of transaction respectively.
- SafeMath is used to handle integer related errors

2. Library: SafeMath

Safemath library provides the safe use signed/unsigned numbers in solidity such as overflow in addition, underflow in subtraction and multiplication and division.

1. Demo

1. Let's assume three accounts in this scenario:
    a. 0x7B3B3B38670A4390B06B569E5a0509a8C16Ab8bb
    b. 0xBe5F751F5b03F5e5Be2e1b00840F705405478280
    c. 0x3d7a6ee3f43C66C398A1216733c5E67edDF4b43D
2. Compile smart contract "MyToken.sol
3. Deploy "MyToken.sol" with name, symbol, decimal point and number of tokens. on account a – [refer to figure 1]



ENVIRONMENT: JavaScript VM
ACCOUNT: 0xedb...2Caa4 (99.999999
GAS LIMIT: 3000000
VALUE: 0 wei
CONTRACT: MyToken - browser/MyToken.so
_NAME: MyToken
_SYMBOL: DXY
_DECIMALS: 0
_TOTALSUPPLY: 100

```
pragma solidity >=0.4.0 <0.7.0;

contract MyToken {

    string public  name ;
    string public  symbol;
    uint256 public  decimals ;

    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
    event Transfer(address indexed from, address indexed to, uint tokens);

    mapping(address => uint256) balances;

    mapping(address => mapping (address => uint256)) allowed;

    uint256 totalSupply_;

    using SafeMath for uint256;

    constructor(string memory _name, string memory _symbol, uint256 _decimals, uint256 _totalSupply) public {

        symbol = _symbol;
        decimals = _decimals;
        name = _name;
        totalSupply_ = _totalSupply  * 10 ** decimals;
        balances[msg.sender] = totalSupply_;
    }

    function totalSupply() public view returns (uint256) {
    return totalSupply_;
    }

    function balanceOf(address tokenOwner) public view returns (uint) {
        return balances[tokenOwner];
    }

    function transfer(address receiver, uint numTokens) public returns (bool) {
        require(numTokens <= balances[msg.sender]);
        balances[msg.sender] = balances[msg.sender].sub(numTokens);
        balances[receiver] = balances[receiver].add(numTokens);
```

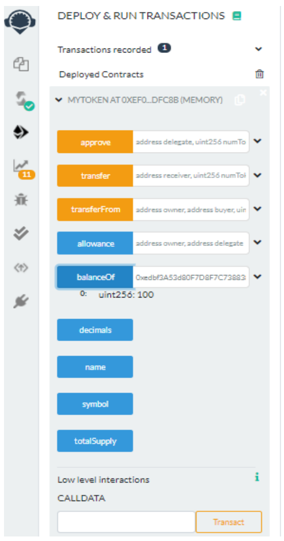
```
[vm] from: 0xedb...2caa4 to: MyToken.(constructor) value: 0 wei data: 0x608...00000 logs: 0 hash: 0x126...8de9f

call to MyToken.balanceOf

CALL [call] from: 0xedbf3A53d80F7D8F7C738838A01626AABdE2Caa4 to: MyToken.balanceOf(address) data: 0x70a...2caa4
```

Fig.1. Smart contract deployed

4. Now accounts having following status
    a. {"0x7B3B3B38670A4390B06B569E5a0509a8C16Ab8bb", balance : "100"
    b. {"0xBe5F751F5b03F5e5Be2e1b00840F705405478280", balance : "0"}
    c. {"0xBe5F751F5b03F5e5Be2e1b00840F705405478280", balance : "0"}


DEPLOY & RUN TRANSACTIONS

Transactions recorded 1

Deployed Contracts

MYTOKEN AT 0XEF0...DFC8B (MEMORY)

approve: address delegate, uint256 numTo

transfer: address receiver, uint256 numTok

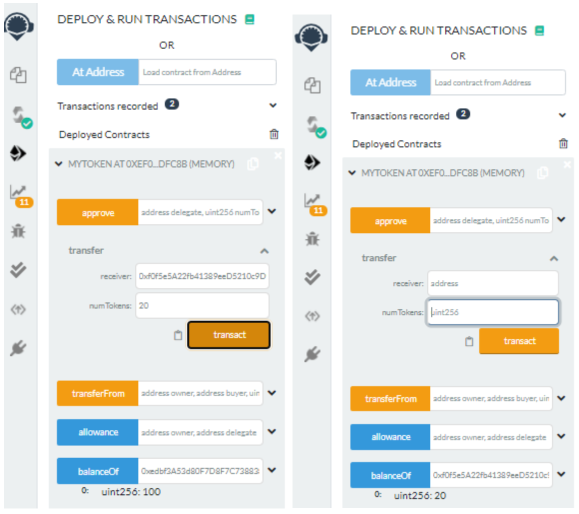
transferFrom: address owner, address buyer, uin

allowance: address owner, address delegate

balanceOf: 0xedbf3A53d80F7D8F7C73883

0: uint256: 100

Fig.2. Balance in account a


DEPLOY & RUN TRANSACTIONS

OR

At Address: Load contract from Address

Transactions recorded 2

Deployed Contracts

MYTOKEN AT 0XEF0...DFC8B (MEMORY)

approve: address delegate, uint256 numTo

transfer

receiver: 0xf0F5e5A22fb41389eeD5210c9D

numTokens: 20

transact

transferFrom: address owner, address buyer, uin

allowance: address owner, address delegate

balanceOf: 0xedbf3A53d80F7D8F7C73883

0: uint256: 100

DEPLOY & RUN TRANSACTIONS

OR

At Address: Load contract from Address

Transactions recorded 2

Deployed Contracts

MYTOKEN AT 0XEF0...DFC8B (MEMORY)

approve: address delegate, uint256 numTo

transfer

receiver: address

numTokens: uint256

transact

transferFrom: address owner, address buyer, uin

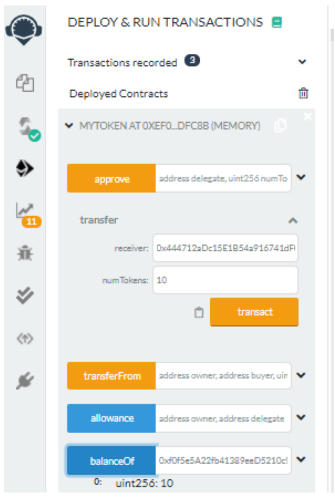
allowance: address owner, address delegate

balanceOf: 0xf0F5e5A22fb41389eeD5210c!

0: uint256: 20

Fig.3. Token transfer from a to b

5. Check balance of account a as shown in figure 2.
6. Token Exchange: Case 1
   a. Execute transfer function that result in Transfer of 20 token from a to b as shown in fig.3
      - {“0xedbf3A53d80F7D8F7C738838A01626AABdE2Caa4”, balance: “80”
      - {“0xf0f5e5A22fb41389eeD5210c9D9203Ea65EF1082”, balance: “20”}
      - {“0x444712aDc15E1B54a916741dF032d9380C45b561”, balance: “0”}
7. Token Exchange: Case 2
   a. Select account b from list of account
   b. Execute transfer function that result in Transfer of 10 token from b to c as shown in fig.4
      - {“0xedbf3A53d80F7D8F7C738838A01626AABdE2Caa4”, balance: “80”
      - {“0xf0f5e5A22fb41389eeD5210c9D9203Ea65EF1082”, balance: “10”}
      - {“0x444712aDc15E1B54a916741dF032d9380C45b561”, balance: “10”}
8. Token Exchange: Case 3
   a. Select account c from list of account
   b. Execute transfer function that result in Transfer of 5 token from c to a as shown in fig.5
      - {“0xedbf3A53d80F7D8F7C738838A01626AABdE2Caa4”, balance: “85”
      - {“0xf0f5e5A22fb41389eeD5210c9D9203Ea65EF1082”, balance: “10”}
      - {“0x444712aDc15E1B54a916741dF032d9380C45b561”, balance: “5”}


DEPLOY & RUN TRANSACTIONS

Transactions recorded 3

Deployed Contracts

MYTOKEN AT 0XEF0...DFC8B (MEMORY)

approve: address delegate, uint256 numTo

transfer

receiver: 0x444712aDc15E1B54a916741dF

numTokens: 10

transact

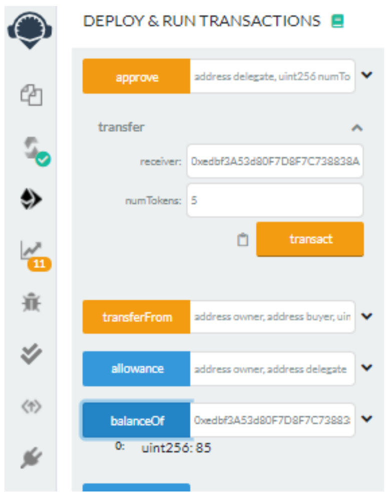
transferFrom: address owner, address buyer, uin

allowance: address owner, address delegate

balanceOf: 0xf0F5e5A22fb41389eeD5210c!

0: uint256: 10

Fig.4. Token transfer from b to c


DEPLOY & RUN TRANSACTIONS

approve: address delegate, uint256 numTo

transfer

receiver: 0xedbf3A53d80F7D8F7C738838A

numTokens: 5

transact

transferFrom: address owner, address buyer, uin

allowance: address owner, address delegate

balanceOf: 0xedbf3A53d80F7D8F7C73883

0: uint256: 85

Fig.5. Token transfer from c to a